

# **ARHITECTURA SISTEMELOR DE CALCUL - CURS 0x05**

**ÎNMULȚIREA/ÎMPĂRȚIREA NUMERELOR ÎNTREGI,  
REPREZENTAREA ÎN VIRGULĂ MOBILĂ**

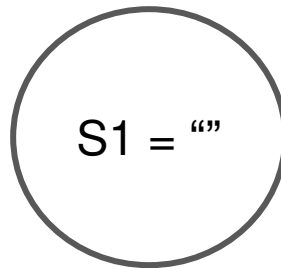
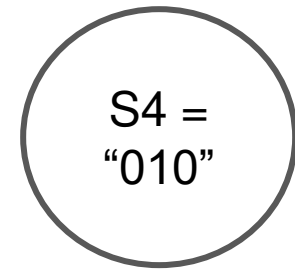
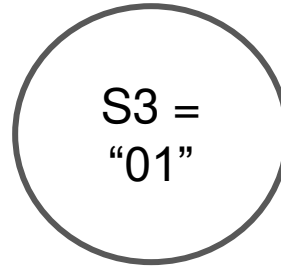
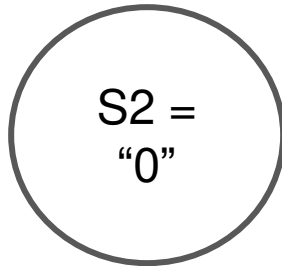
Cristian Rusu

# CUPRINS

- **logică secvențială + combinatorială, un exemplu**
- **înmulțirea numerelor întregi binare**
- **împărțirea numerelor întregi binare**
- **reprezentarea numerelor în virgulă mobilă**
- **lucrul cu numerele în virgulă mobilă**

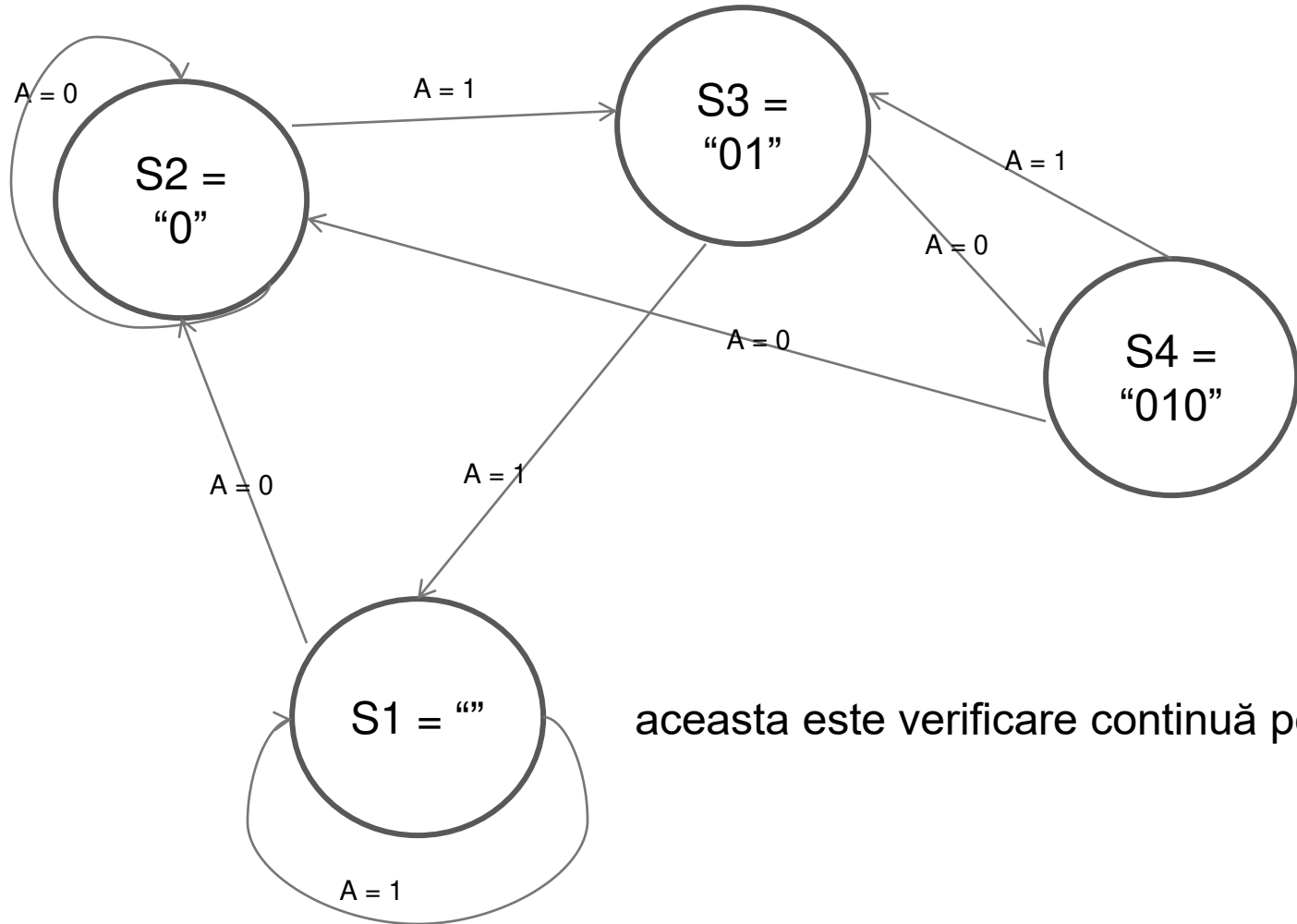
# SECVENȚIAL, SEMINAR 0x02, EX 10

- un semnal digital A poate lua valori {0, 1} în timp iar noi vrem să detectăm dacă semnalul are valoarea 010 la un moment dat. Dacă această secvență de biți este detectată în A atunci o variabilă Y este setată la 1, altfel această variabilă este 0
- definim 4 stări



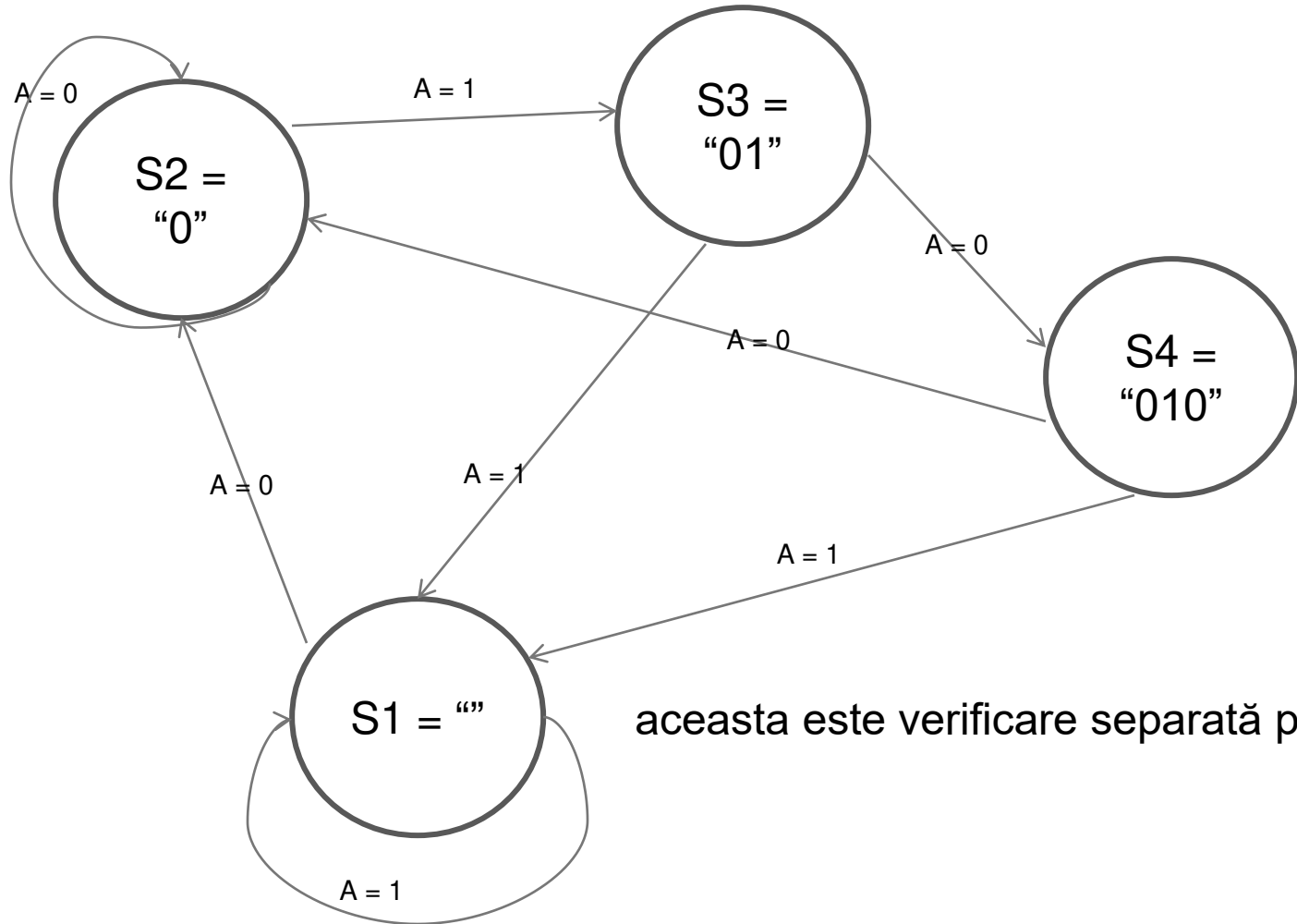
care sunt tranzițiile între aceste stări?

# SECVENȚIAL, SEMINAR 0x02, EX 10



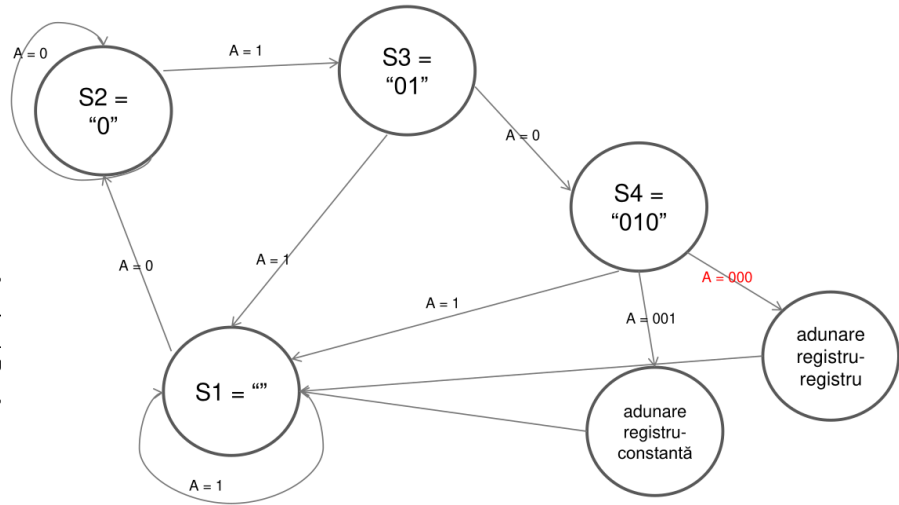
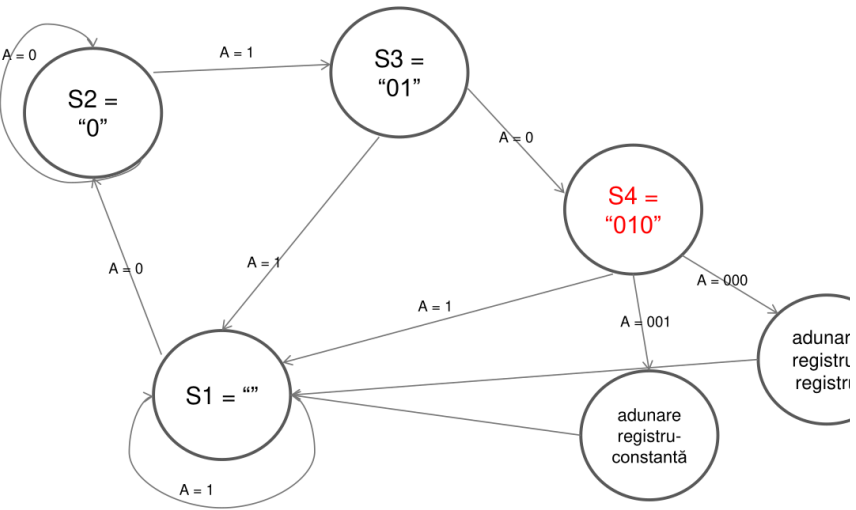
aceasta este verificare continuă pentru 010

# SECVENȚIAL, SEMINAR 0x02, EX 10

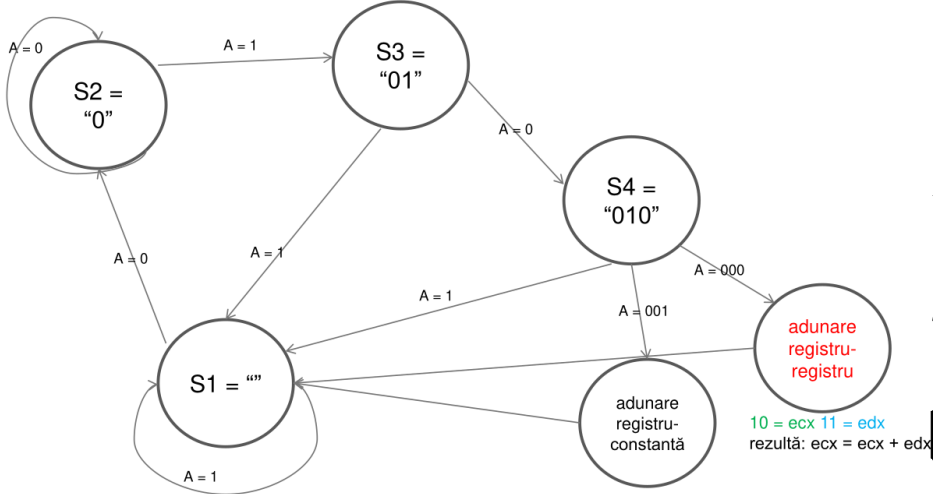


aceasta este verificare separată pentru 010

# CE FACE UN PROCESOR

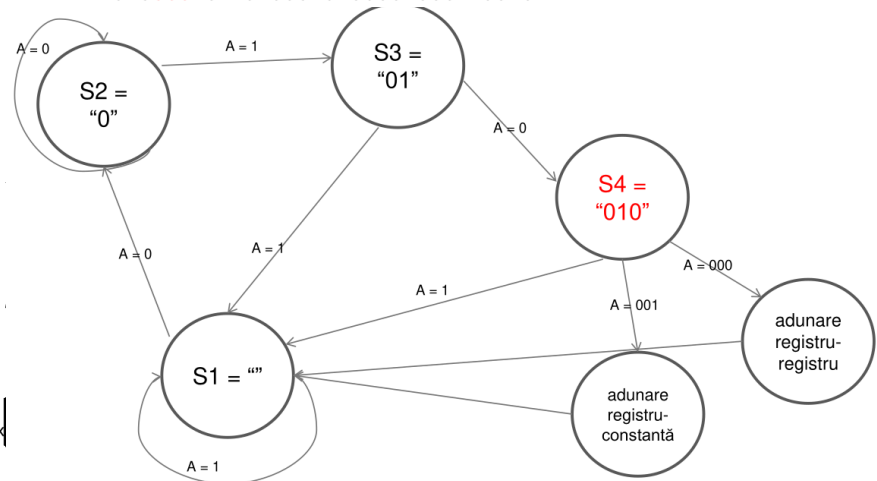


...0100001011010001010000100011001011 ...



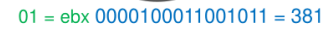
...0100001011010001010000100011001011 ...

...010**000**1011010001010000100011001011 ...



...0100001011**01**0001010000100011001011 ...

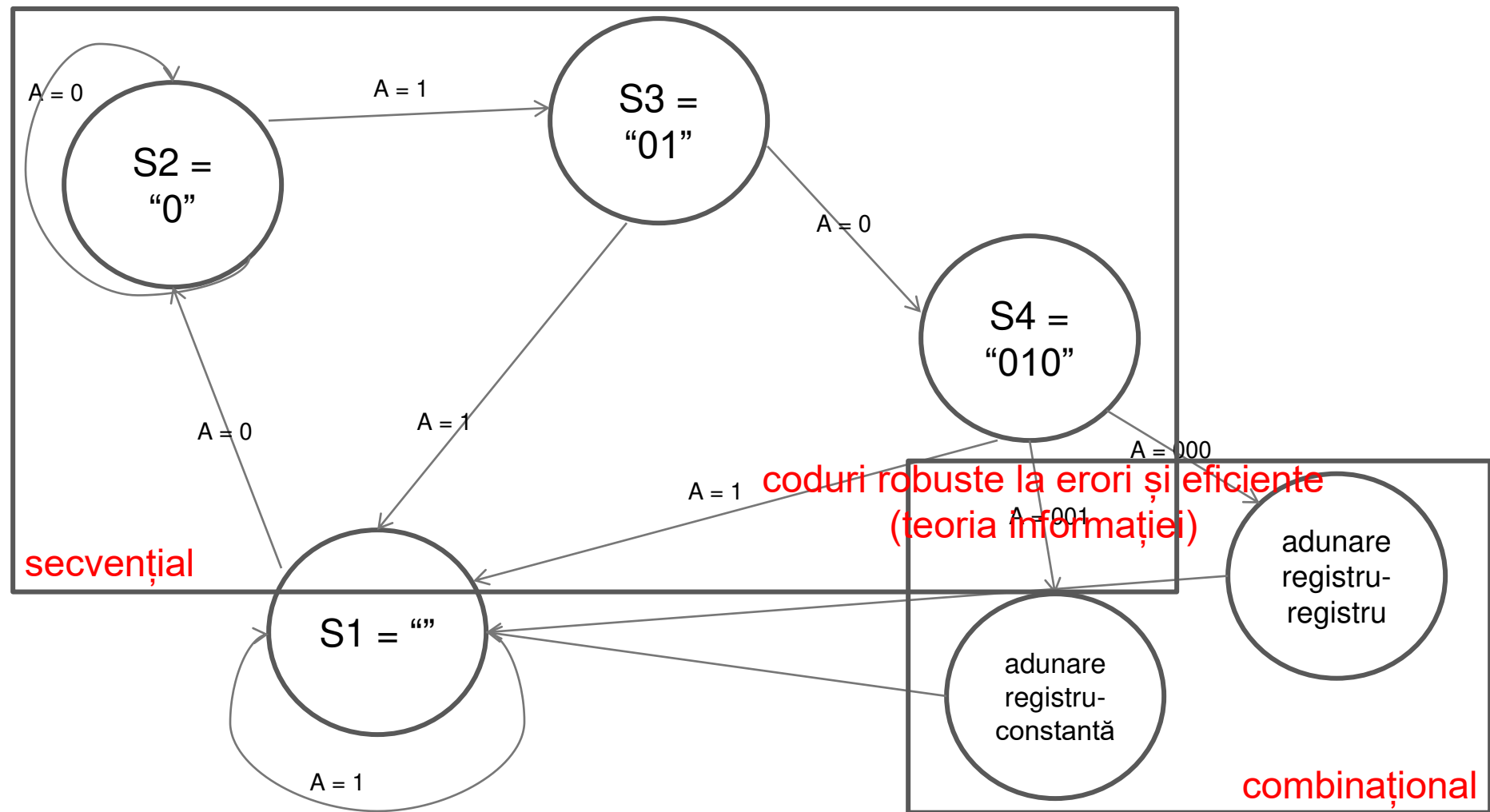
**QUESTION**



...0100001011010001010000100011001011 ...



# CE FACE UN PROCESOR



cod mașină ...0100001011010001010000100011001011 ...



# CONȚINUT NOU PENTRU CURS

- **înmulțirea numerelor întregi binare**
- **împărțirea numerelor întregi binare**
- **reprezentarea numerelor în virgulă mobilă**
- **lucrul cu numerele în virgulă mobilă**

# ÎNMULȚIREA NUMERELOR ÎNTREGI

- exemplu,  $s = a \times b$

- $a$  și  $b$  pe  $N$  biți
- $s$  pe  $2N$  biți

1	1	1	0
---	---	---	---

 $a$ 

0	1	0	1
---	---	---	---

 $b$ 

---

 $s$

# ÎNMULȚIREA NUMERELOR ÎNTREGI

- exemplu,  $s = a \times b$

1 1 1 0	$a = 14$
0 1 0 1	$b = 5$
<hr/>	
1 1 1 0	$s = 70$
0 0 0 0	
1 1 1 0	
0 0 0 0	
<hr/>	
1 0 0 0 1 1 0	

ce am făcut aici este corect,  
dar am presupus că am primit  
numere naturale. ce se  
întâmplă dacă  $a$  și  $b$  sunt în  
complement față de doi?

# ÎNMULȚIREA NUMERELOR ÎNTREGI

- exemplu,  $s = a \times b$

1	1	1	0
---	---	---	---

$a = -2$

0	1	0	1
---	---	---	---

$b = 5$

---

$s = -10$

primul pas: extindem operanzii  
pe 8 biți

# ÎNMULȚIREA NUMERELOR ÎNTREGI

- exemplu,  $s = a \times b$

1 1 1 1 1 1 1 0
-----------------

 $a = -2$ 

0 0 0 0 0 1 0 1
-----------------

 $b = 5$ 

---

1 1 1 1 1 1 1 0
-----------------

 $s = -10$ 

0 0 0 0 0 0 0 0
-----------------

1 1 1 1 1 1 1 0
-----------------

...
-----

---

.....0 1 1 1 1 0 1 1 0
------------------------

al doilea pas: facem operația  
de înmulțire obișnuită

# ÎNMULȚIREA NUMERELOR ÎNTREGI

- exemplu,  $s = a \times b$

1 1 1 1 1 1 1 0
-----------------

 $a = -2$ 

0 0 0 0 0 1 0 1
-----------------

 $b = 5$ 

---

1 1 1 1 1 1 1 0
-----------------

 $s = -10$ 

0 0 0 0 0 0 0 0
-----------------

1 1 1 1 1 1 1 0
-----------------

...
-----

---

.....0 1 1 1 1 0 1 1 0
------------------------

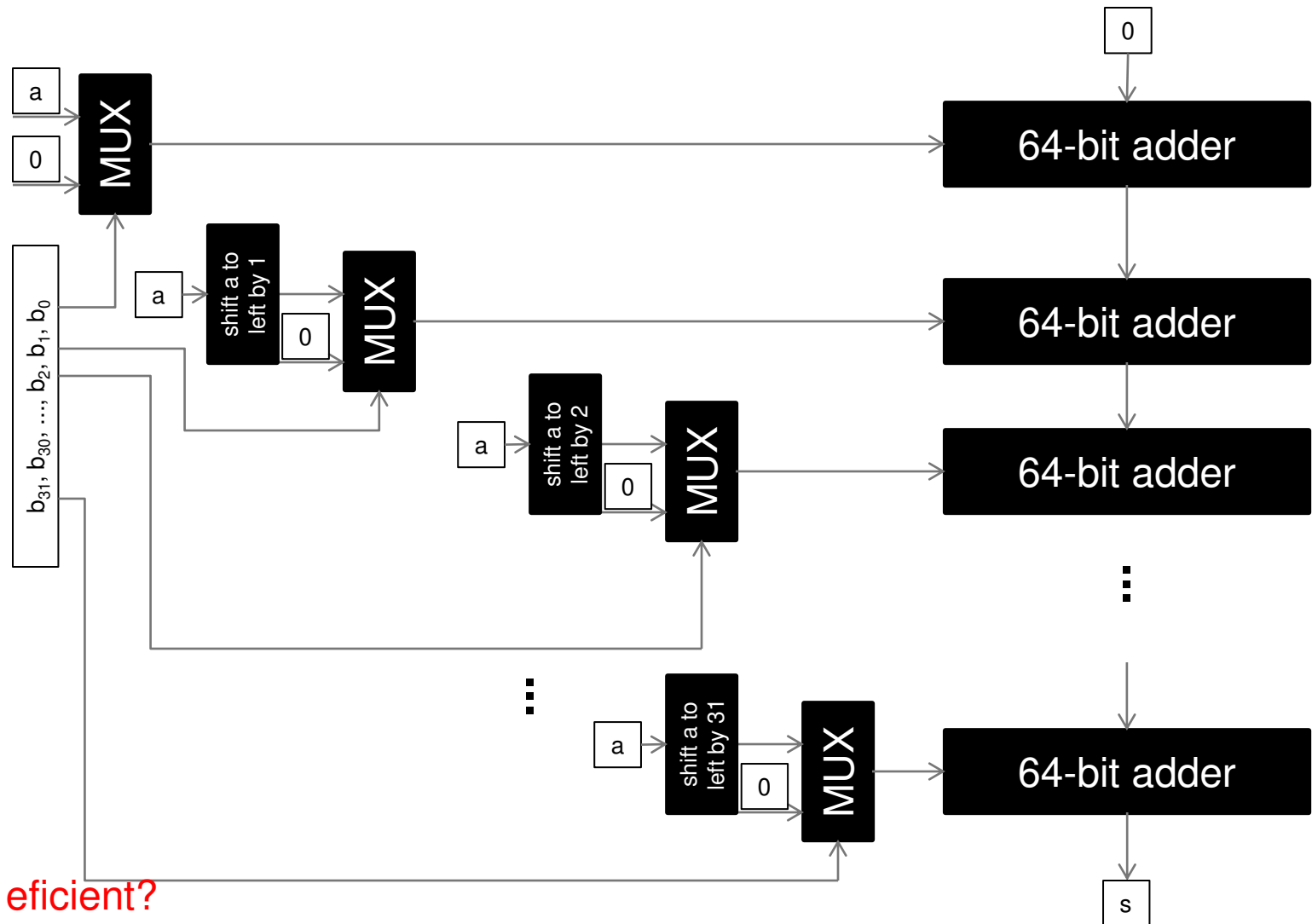
al treilea pas: rezultatul este pe  
8 biți în complement față de doi

# ÎNMULȚIREA NUMERELOR ÎNTREGI

- **circuitul combinațional**
  - $s = a \times b$ ,  $a$  și  $b$  sunt numere pe 32 de biți
  - cum facem?

# ÎNMULȚIREA NUMERELOR ÎNTREGI

- **circuitul combinațional**
  - $s = a \times b$ ,  $a$  și  $b$  sunt numere pe 32 de biți

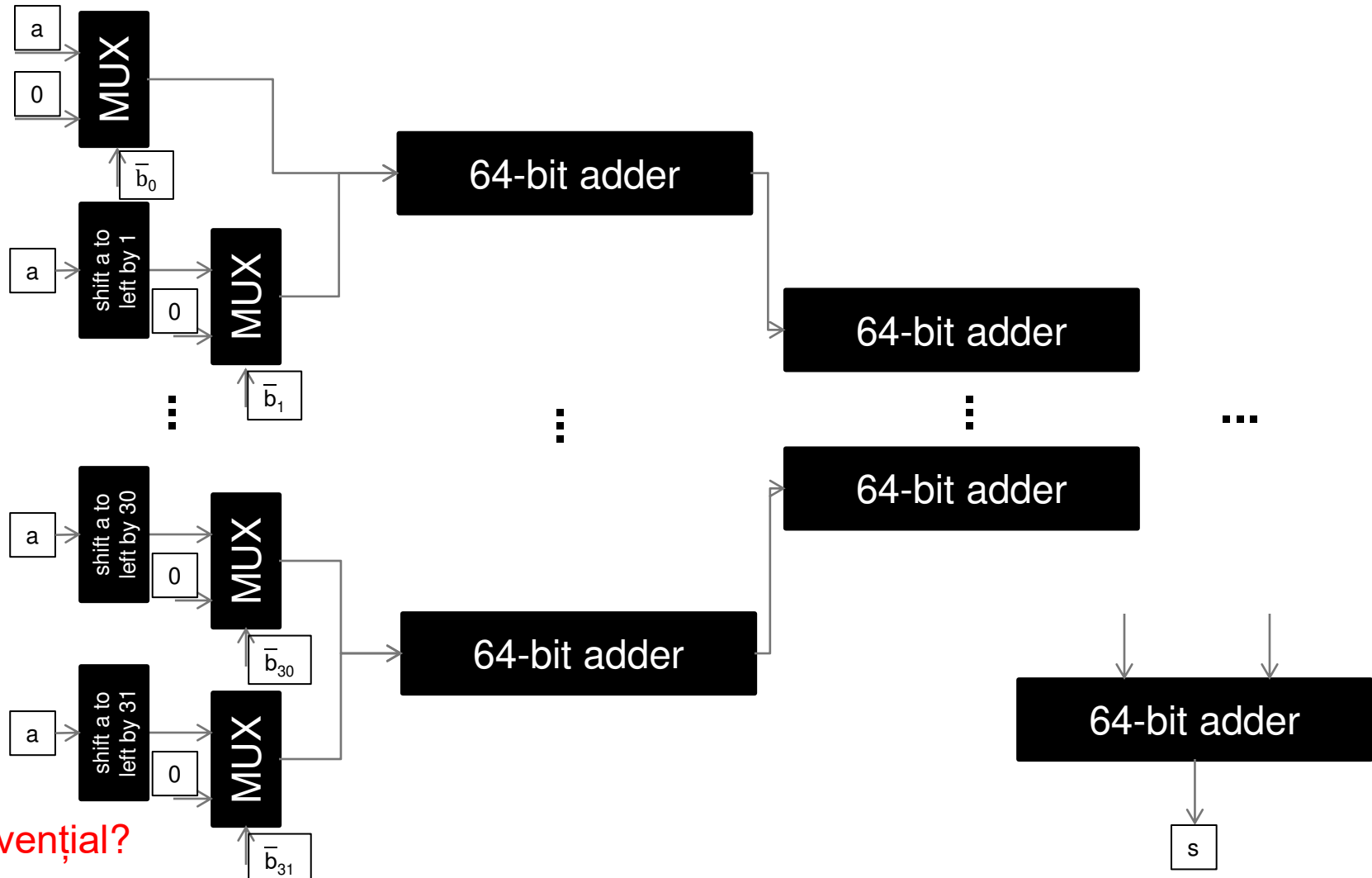


ceva mai eficient?



# ÎNMULȚIREA NUMERELOR ÎNTREGI

- circuitul combinațional
  - $s = a \times b$ ,  $a$  și  $b$  sunt numere pe 32 de biți

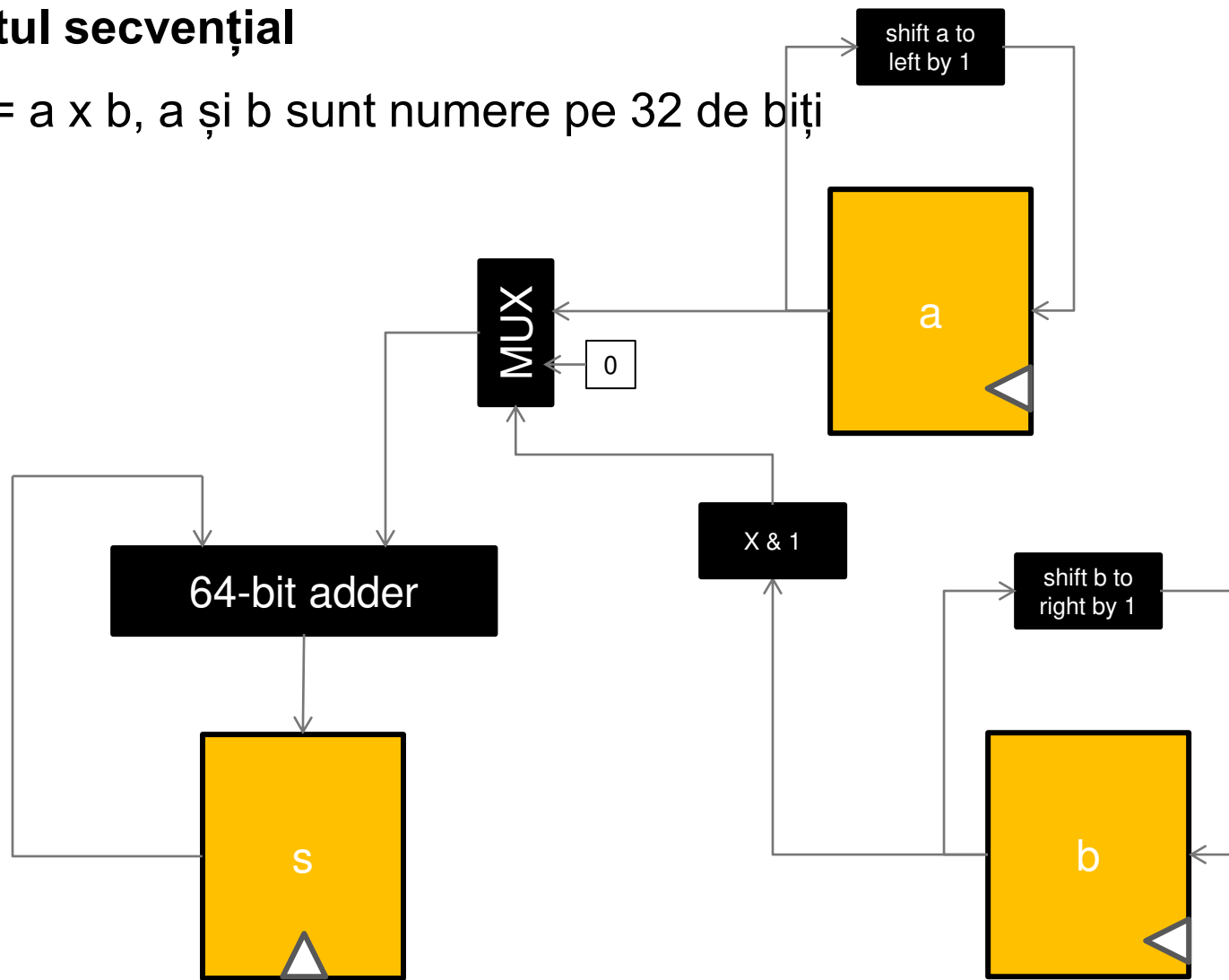


ceva secvențial?

# ÎNMULȚIREA NUMERELOR ÎNTREGI

- **circuitul secvențial**

- $s = a \times b$ ,  $a$  și  $b$  sunt numere pe 32 de biți



# ÎMPĂRȚIREA NUMERELOR ÎNTREGI

- exemplu,  $s = a \div b$

1 0 0 1 1 1

1 1

1 0 0 1 1 1

1 1

0

1 0 0 1 1 1

1 1

0 0

1 0 0 1 1 1

1 1

0 0 1

1 1 1 1

1 1

0 0 1

1 1 1 1

1 1

0 0 1 1

1 1

1 1

0 0 1 1 0

1 1

1 1

0 0 1 1 0 1

$a = 39$

$b = 3$

$s = 13$

# ÎMPĂRȚIREA NUMERELOR ÎNTREGI

- $s = a \div b$ 
  - ce se întâmplă dacă  $a$  sau  $b$  sunt variabile negative?
  - rezultatul este negativ dacă  $a$  și  $b$  au semne diferite (XOR logic)
  - în general
    - $a = s \times b + r$
    - semnul lui  $r$  este semnul lui  $a$
- circuitul pentru împărțire nici nu vom încerca să îl facem
- din cauza acestei complexități ridicate, compilatoarele și sistemele de calcul vor face tot posibilul pentru a evita o împărțire
- vedem mai multe exemple la seminar ...

# REPREZENTAREA ÎN VIRGULĂ MOBILĂ

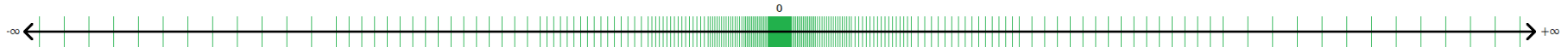
- am discutat la Seminar 0x00 despre reprezentarea în virgulă fixă

...	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$		$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$	$2^{-5}$	$2^{-6}$	$2^{-7}$	...
-----	-------	-------	-------	-------	-------	-------	-------	-------	--	----------	----------	----------	----------	----------	----------	----------	-----

- exemplu: 7.5 e scris ca 111.1
- care este problema cu această reprezentare?
  - partea întreagă este separată de partea fracționară
  - fiecare are nevoie de un număr de biți prestabilit
  - asta poate să fie inefficient
  - vrem ca numărul de biți total să fie alocat “dinamic”, în funcție de numărul pe care trebuie să îl reprezentăm

# REPREZENTAREA ÎN VIRGULĂ MOBILĂ

- **când trebuie să reprezentăm un număr real**
  - nu putem să avem precizie infinită
  - avem un număr finit de biți, deci putem să scriem biții în circuite
  - avem nevoie de precizie variabilă
  - putem avea precizie “infinită” dacă avem numere raționale (și vom salva separat numărătorul și numitor ca întregi)
- **standardul: IEEE 754 Floating Point**
  - densitatea nu este uniformă pe linia reală



# REPREZENTAREA ÎN VIRGULĂ MOBILĂ

- **standardul: IEEE 754 Floating Point**
  - densitatea nu este uniformă pe linia reală



- **sunt câteva consecințe**
  - $(0.1 + 0.2) == 0.3$  versus  $(0.2 + 0.3) == 0.5$  (rotunjiri)
  - $\text{math.sqrt}(3) * \text{math.sqrt}(3) == 3$  versus  $\text{math.sqrt}(3*3) == 3$
  - $(0.7 + 0.2) + 0.1$  versus  $(0.7 + 0.1) + 0.2$  (nu avem asociativitatea)
  - diferența cu numere întregi
    - dacă folosim tip de date întreg:  $16777216 + 1 = 16777217$
    - dacă folosim tip de date FP:  $16777216.0 + 1 = 16777216.0$
    - $\text{float}(123456789101112) + 1.0 = 123456789101113.0$
    - $\text{float}(1234567891011121) + 1.0 = 1234567891011122.0$
    - $\text{float}(12345678910111213) + 1.0 = 1.2345678910111212\text{e}+16$

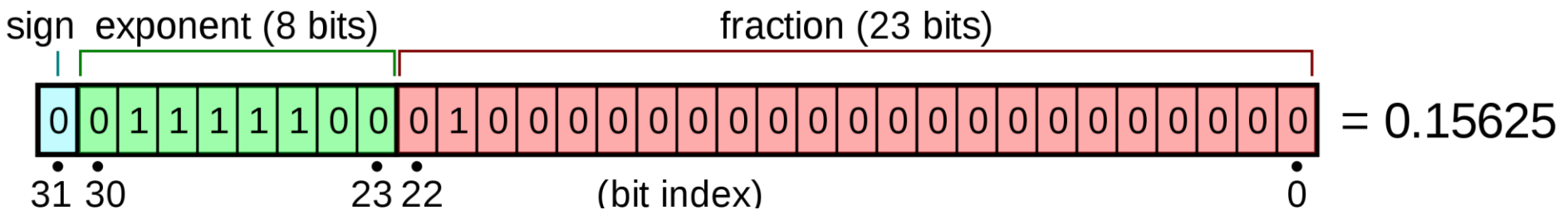
# REPREZENTAREA ÎN VIRGULĂ MOBILĂ

- reprezentarea științifică
  - $12345 = 1.2345 \times 10^4$
  - $5024 = 5.024 \times 10^3$
  - $0.00925 = 9.25 \times 10^{-3}$
- $\text{float}(12345678910111213) + 1.0 = 1.2345678910111212\text{e}+16$
- $101010 = 1.01010 \times 2^5$ 
  - în sistemul binar, primul bit din reprezentare este mereu 1



# REPREZENTAREA ÎN VIRGULĂ MOBILĂ

- **standardul: IEEE 754 Floating Point**



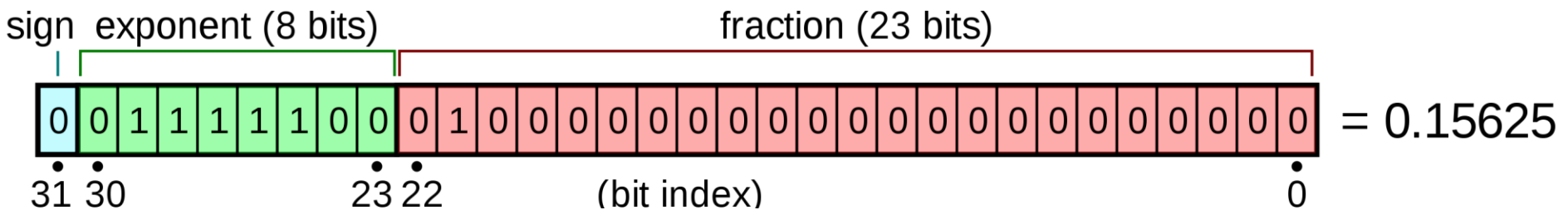
- $x = (-1)^s \cdot 1.\text{mmmmmmmmmmmmmmmmmmmmmmmm} \cdot 2^{(\text{eeeeeee})_2 - 127}$

- **example:**

- $0.15625 = (-1)^0 1.0100...0 2^{b01111100 - 127} = 1.25 2^{-3} = 1.25/8$
- alte exemple:
- $(-1)^0 1.1000...0 2^{b01111100 - 127} = 1.5 2^{-3} = 1.5/8 = 0.1875$
- $1 = (-1)^0 1.000000000000000000000000 2^{b01111111 - 127}$
- $-1 = (-1)^1 1.000000000000000000000000 2^{b01111111 - 127}$
- $2 = (-1)^0 1.000000000000000000000000 2^{b10000000 - 127}$
- $\infty = (-1)^0 1.000000000000000000000000 2^{b11111111 - 127}$
- $-\infty = (-1)^1 1.000000000000000000000000 2^{b11111111 - 127}$

# REPREZENTAREA ÎN VIRGULĂ MOBILĂ

- **standardul: IEEE 754 Floating Point**



- $x = (-1)^s 1.\text{mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm} 2^{(\text{eeeeeeee})_2 - 127}$
- **exemple:**
  - $\pi \approx 3.14159265 = (-1)^0 1.10010010000111111011010 2^{b10000000 - 127}$
  - $+0 = (-1)^0 1.0000000000000000000000000000 2^{00000000 - 127}$
  - $-0 = (-1)^1 1.0000000000000000000000000000 2^{00000000 - 127}$
  - signaling NaN: 0x7F800001 sau 0x7FBFFFFFFF sau între 0xFF800001 și 0xFFBFFFFFFF
  - quiet NaN: 0x7FC00000 sau 0x7FFFFFFF sau între 0xFFC00000 și 0xFFFFFFFFFF

mai mult exemple, la Seminarul 0x03